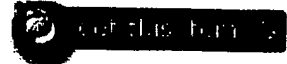


AVAILABLE COPY

PRINT WINDOW CLOSE WINDOW



◀ next ▶

**PAJ 07-01-01 01202397 JP ARCHITECTURE DESIGN SUPPORTING SYSTEM FOR  
SYSTEM-ON-CHIP AND ARCHITECTURE GENERATING METHOD**

**INVENTOR(S)- NISHI, HIROAKI**

**PATENT APPLICATION NUMBER- 2000012047**

**DATE FILED- 2000-01-20**

**PUBLICATION NUMBER- 01202397 JP**

**DOCUMENT TYPE- A**

**PUBLICATION DATE- 2001-07-27**

**INTERNATIONAL PATENT CLASS- G06F01750; H01L02182**

**APPLICANT(S)- TOSHIBA CORP**

**PUBLICATION COUNTRY- Japan NDN- 043-0218-3246-5**

**PROBLEM TO BE SOLVED:** To provide an architecture design supporting system for system-on-ship and an architecture generating method, with which man-hour for system architecture design is reduced. **SOLUTION:** The method supports the design of the system architecture of a system-on-ship from the description of the specification of a system to become an object. By using recyclable flexible hardware(HW) and software (SW) modules from the previously applied specification non-differentiating HW and SW, the design of HW and SW architectures can be supported. **COPYRIGHT: (C)2001,JPO**

NO-DESCRIPTORS

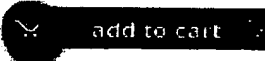
---

[back to top](#)

Copyright Fee: \$ 0.00

Document Price: \$ 13.50

**Total: \$ 13.50\***



\* If this item is not available from our original source at the price quoted you will be notified.  
Shipping by US 1st Class Mail or eDoc is free of charge.

Nerac, Inc. One Technology Drive . Tolland, CT  
Phone (860) 872-7000 . Fax (860) 875-1749  
©1995-2003 All Rights Reserved.

**BEST AVAILABLE COPY**

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2001-202397

(P2001-202397A)

(43) 公開日 平成13年7月27日 (2001.7.27)

(51) Int. Cl.	識別記号	F I	テームト (参考)
G 0 6 F 17/50		G 0 6 F 15/60	6 5 4 M 5 B 0 4 6
H 0 1 L 21/82		H 0 1 L 21/82	C 5 F 0 6 4

審査請求 未請求 請求項の数 6 O L (全 13 頁)

(21) 出願番号 特願2000-12047 (P2000-12047)

(22) 出願日 平成12年1月20日 (2000.1.20)

(71) 出願人 000003078

株式会社東芝

神奈川県川崎市幸区堀川町72番地

(72) 発明者 西 宏晃

神奈川県川崎市幸区小向東芝町1番地 株

式会社東芝マイクロエレクトロニクスセン

ター内

(74) 代理人 100083808

弁理士 三好 秀和 (外7名)

Fターム (参考) 5B04B AA06 BA02 KA06

5F06A AA10 DD03 DD07 EED3 EE47

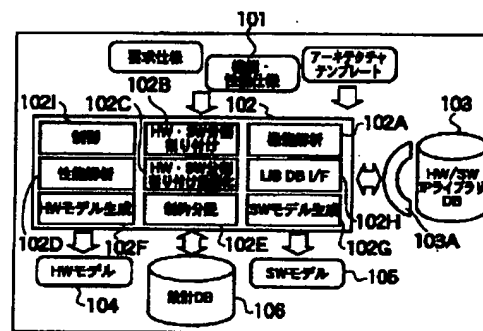
HH05 HH06 HH08 HH12

(54) 【発明の名称】 システム・オン・チップのアーキテクチャ設計支援システム及びアーキテクチャ生成方法

(57) 【要約】

【課題】 システムアーキテクチャ設計工数が小さいシステム・オン・チップのアーキテクチャ設計支援システム及びアーキテクチャ生成方法を提供する。

【解決手段】 対象となるシステムの仕様の記述から、システム・オン・チップのシステム・アーキテクチャの設計を支援する方法が示されている。予め与えられているハードウェアとソフトウェアが未分化な仕様から、再利用可能なフレキシブルなHWとSWモジュールを利用することにより、HW、SWアーキテクチャの設計を支援することを可能とした。



【特許請求の範囲】

【請求項1】 対象となるシステムの仕様の記述から、システム・オン・チップのシステム・アーキテクチャの設計を支援するシステムであって、ハードウェアモジュール及びソフトウェアモジュールの再利用可能なライブラリと、前記仕様記述に対応する複数のタスクの夫々に関して選択され、その実現に適したハードウェアモジュールとソフトウェアモジュールの組み合わせから、前記仕様記述に対応するシステムを実装するためのハードウェア及びソフトウェアのシステム・アーキテクチャを生成することを特徴とするシステム・オン・チップのアーキテクチャ設計支援システム。

【請求項2】 対象となるシステムの仕様の記述から、システム・オン・チップのシステム・アーキテクチャの設計を支援する方法であって、前記仕様記述に対応する複数のタスクの夫々に関して、その実現に適したハードウェアモジュールとソフトウェアモジュールの組み合わせを、ハードウェアモジュール及びソフトウェアモジュールの再利用可能なライブラリから選択するステップと、前記ハードウェアモジュールと前記ソフトウェアモジュールの組み合わせから、前記仕様記述に対応するシステムを実装するためのハードウェア及びソフトウェアのシステム・アーキテクチャを生成するステップからなることを特徴とするシステム・オン・チップのアーキテクチャ生成方法。

【請求項3】 前記ハードウェアモジュールと前記ソフトウェアモジュールの組み合わせの選択は、サイズ、遅延、消費電力、テストコストのいずれか1つを最適化する様に行われることを特徴とする請求項2に記載のアーキテクチャ生成方法。

【請求項4】 前記ハードウェアモジュールと前記ソフトウェアモジュールの組み合わせの選択は、フレキシブルな抽象アーキテクチャのテンプレートを用いて行われることを特徴とする請求項2に記載のアーキテクチャ生成方法。

【請求項5】 前記システムの仕様は、状態遷移と各状態で実行されるタスクをプロセスに持ち、このプロセスの集まりをブロックとしてさらにこのブロック同士のチャネルで表現されることを特徴とする請求項2に記載のアーキテクチャ生成方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、仕様記述からのシステム・オン・チップのシステム・アーキテクチャの設計を支援する為のアーキテクチャ生成方法に関する。特に、本発明は、システム・オン・チップのハードウェア(HW)とソフトウェア(SW)のアーキテクチャ設計支援方法を提案するものであり、ハードウェアとソフトウェアが未分化な仕様から再利用可能でフレキシブルなHWとSWモジュールを利用してHW、SWアーキテク

チャの設計を支援する方法に関する。

【0002】

【従来の技術】 プロセス技術の進歩によってLSIの集積度が飛躍的に増大し、これまでボードで実現していたシステムを、システム・オン・チップ(SoC)として1チップの上に搭載することができるようになった。

【0003】 チップに搭載する部品も、スタンダードセルやメモリモジュールに加え、CPU (Central Processing Unit) コア、AS (Application Specific) モジュール、DRAM (Dynamic Random Access Memory) モジュール、アナログモジュールなどと多様化している。これに伴い、SoCを効果的に設計するための新しいデザインフロー、および設計環境が必要となってきた。

【0004】 従来のSoCのシステム・アーキテクチャの設計を支援する方法は、機能仕様とアーキテクチャモデルを人がリンクし、機能をアーキテクチャ上でシミュレーションしながらHWとSWのトレードオフを判断しアーキテクチャを設計するという手順を踏んでいる。

【0005】

【発明が解決しようとする課題】 しかしながら、従来の技術では、ハードウェア・アーキテクチャの各モジュールはテンプレートとして用意されるが、モデルが固く使用時にその要素を変えることができないという欠点があった。さらに、機能仕様の構成要素の各ブロックとシステム・アーキテクチャの各ブロックは多対多の関係であるにもかかわらず、1つの機能ブロックをHWブロックに割り当てるだけで、機能ブロックを分割したり、またマージしたりしてHWにマッピングすることは行われていなかった。

【0006】 従って、本発明の目的は、システムアーキテクチャ設計工数の小さいシステム・オン・チップのアーキテクチャ設計支援システム及びアーキテクチャ生成方法を提供することである。

【0007】 本発明の別の目的は、具象化の選択肢が多く、最適アーキテクチャの設計のため広いスペースで設計できるシステム・オン・チップのアーキテクチャ設計支援システム及びアーキテクチャ生成方法を提供することである。

【0008】 本発明の更に別の目的は、仕様から、システムアーキテクチャの設計のギャップが少なく、また支援ツールの実現も容易なシステム・オン・チップのアーキテクチャ設計支援システム及びアーキテクチャ生成方法を提供することである。

【0009】

【課題を解決するための手段】 上記目的を達成するため、本発明(請求項1)は、対象となるシステムの仕様の記述から、システム・オン・チップのシステム・アーキテクチャの設計を支援するシステムであって、ハードウェアモジュール及びソフトウェアモジュールの再利用

可能なライブラリと、前記仕様記述に対応する複数のタスクの夫々に関して選択され、その実現に適したハードウェアモジュールとソフトウェアモジュールの組み合わせから、前記仕様記述に対応するシステムを実装するためのハードウェア及びソフトウェアのシステム・アーキテクチャを生成することを特徴とするシステム・オン・チップのアーキテクチャ設計支援システムを提供する。

【0010】本発明（請求項2）は、対象となるシステムの仕様の記述から、システム・オン・チップのシステム・アーキテクチャの設計を支援する方法であって、前記仕様記述に対応する複数のタスクの夫々に関して、その実現に適したハードウェアモジュールとソフトウェアモジュールの組み合わせを、ハードウェアモジュール及びソフトウェアモジュールの再利用可能なライブラリから選択するステップと、前記ハードウェアモジュールと前記ソフトウェアモジュールの組み合わせから、前記仕様記述に対応するシステムを実装するためのハードウェア及びソフトウェアのシステム・アーキテクチャを生成するステップからなることを特徴とするシステム・オン・チップのアーキテクチャ生成方法を提供する。

【0011】本発明（請求項3）は、上記請求項2において、前記ハードウェアモジュールと前記ソフトウェアモジュールの組み合わせの選択は、サイズ、遅延、消費電力、テストコストを最適化する様に行われることを特徴とするアーキテクチャ生成方法を提供する。

【0012】本発明（請求項4）は、上記請求項2において、前記ハードウェアモジュールと前記ソフトウェアモジュールの組み合わせの選択は、フレキシブルな抽象アーキテクチャのテンプレートを用いて行われることを特徴とするアーキテクチャ生成方法を提供する。

【0013】本発明（請求項5）は、上記請求項2において、前記システムの仕様は、状態遷移と各状態で実行されるタスクをプロセスに持ち、このプロセスの集まりをブロックとしてさらにこのブロック同士のチャネルで表現されることを特徴とするアーキテクチャ生成方法を提供する。

【0014】

【発明の実施の形態】以下、本発明の実施の形態を図面に基いて説明する。ここで説明するシステム・オン・チップのアーキテクチャ設計支援方法は、一般的な情報処理システムによって実現できる。そのハードウェア構成を図1に示す。

【0015】即ち、以下の各実施例で説明する各ステップの処理を行うためのプロセッサを含むコンピュータ1、ハードディスク装置などの内部記憶手段3、マウス5やキーボード6などの入力手段、モニタ7やプリンタ9などの出力手段、及びCD-ROM11やフレキシブルディスク13等の外部記憶媒体及びその駆動ドライブ15からなっている。

【0016】外部記憶媒体には、以下の各実施例で説明

する各ステップの処理を行うための手続きを記述したプログラムが作成され格納されている。このプログラムは、例えば、フォートランやC言語等の高級言語に必要によりアセンブリ言語を組み合わせで作成される。そのようなプログラムを、外部記憶媒体から適宜内部記憶手段3にインストールし、実行時にメインメモリにロードされる。尚、これらは通常の汎用コンピュータシステムを用いてもよい。

【0017】（1）第1実施形態

本来、仕様とは、実装に対する要求機能や動作を表わし、実装モデルとは区別して用いる必要がある。例えば、レジスタ転送レベルのHDL記述も回路の仕様であり、ソフトウェアのC言語のプログラムも仕様と見なされる。なぜなら、HDL記述では、論理合成ツールによって、又、C言語プログラムでは、コンパイラの最適化手段によって、夫々実際に作成される回路モデルやコード列が異なるからである。しかし、実装設計は高位で行われるようになってきており、最上位のシステム定義だけを仕様として、それより下位の設計対象を実装モデルとして以下説明する。ただし、例外としてシステム定義の再利用モジュールをモデルと呼ぶことがある。

【0018】さらに、以下の説明では、再利用可能な機能仕様モジュール、HWモジュール、SWモジュールのそれぞれを、FNS-IP（機能IP）、HW-IP、SW-IPと呼ぶことにする。IPとは一般に知的所有（財産）権を指すが、半導体業界では再利用可能で売買可能なLSIの設計データをIPと呼んでおり、ここではそれを拡張してLSIの機能モデル、HWモデル、SWモデルもIPと呼ぶ。なおモジュールとは外部環境との境界すなわち入出力が明解になっている処理単位のことである。

【0019】FNS-IPは仕様記述言語SDL（Specification and Description Language）などの仕様記述中のタスクで使用される組み込み関数を指す。HW-IPはプロセッサクラス、DSPクラス、バスクラスなどの抽象クラスのHWから最下位クラスの特定プロセッサやバスを表すHDL記述である。またHW-IPの各階層のモデルはHDLもしくはC言語で記述され、データ幅などがパラメタライズ可能な記述である。SW-IPはプログラム言語で表現されたアルゴリズムの構成要素で、例えば様々なコードの符号、復号処理などのあらかじめ定義された関数の記述である。HW-IPとSW-IPは各種パラメータ（IPとIP内合成パラメータ）を持つ。HW-IPパラメータには、HW-IP名、カテゴリ（処理IP、コミュニケーションIP、記憶IP）、クラス（CPU、専用HW、インターフェース、メモリ）がある。また、各クラスには、以下の様なタイプがある。

【0020】CPU…標準CPU、構成可変CPU、標準DSP、構成可変DSP

専用HW…コプロセッサ、ASIC、FPGA

インターフェース…バス（種類、階層、プロトコル）、  
バッファ、バスコントローラ、ブリッジ

メモリ…SRAM、DRAM、ROM、EPPROM  
一方、HW-IP内パラメータとしては、以下の様なタイプがある。

【0021】CPU…ワード数、命令形式タイプ、各種レジスタ数、レジスタファイルサイズ、キャッシュサイズ他

専用HW…ポート数、ポートビット幅

インターフェース…バス幅、制御方式

メモリ…ポート数、ワード数、バンク数

又、SW-IPパラメータにはSW-IP名、カテゴリ（処理IP、HW/SWインターフェース、OS）、クラス（アプリケーション、デバイスドライバ、ミドルウェア）がある。また内部パラメータとしては、各SW-IPが処理するデータサイズや、SW-IP自体のコード量すなわちワード数（プロセッサ名とワードサイズをパラメータにして）がある。

【0022】まず、図2を使用して本発明のアーキテクチャ設計支援方法の構成手段を説明し、そのあと各構成手段の処理内容を説明していく。

【0023】HW/SW合成部102は、HW/SW-IPライブラリデータベース管理システム（HW/SW-IP〜DBMS）103A、103からHW-IPやSW-IPを取り込み、機能・性能仕様101を要求仕様の下でHWモデル104とSWモデル105とに合成する。

【0024】機能・性能仕様101はSoCを実装ターゲットにした機能と性能仕様で、たとえばSDLのような並行プログラム言語で記述される。これは機能仕様をプロセス毎に表現したもので、タスクが処理の単位である。図3に機能仕様をSDLで記述した例を示す。SDLではブロックで構造が表現でき、ブロック内の動作をプロセスで表現する。ブロックは並列実行され、ブロック間の信号はチャンネルと称し遅延を持つ。一方、プロセス間の信号はルートと称され遅延は持たない。図4にプロセス記述のテキストの例を示す。プロセスは状態遷移で表現され、入力信号がトリガされたらその状態が活性化されタスクが実行される。タスクには代入文が記述されており、右辺の演算結果が左辺に代入され、その右辺には定義済みの関数や演算が記述されている。

【0025】性能仕様はプロセスはブロックに与えられる制約で、例えばプロセスAはレイテンシからm（n、mは整数）との間に実行される、またプロセスAとプロセスBの実行開始時刻制約がTA（msecまたはnsec）、TB（msecまたはnsec）等である。さらに各プロセスまたはブロックが処理に要するメモリ、各プロセスやブロック間でのスループットの要求を記述する（図7）。

【0026】HWやSWの実装にかかわる要求仕様としては、後に詳細に説明するように、SoCのサイズ、動作速度、消費電力、テストコストを指定する。

【0027】これらの機能、性能、要求仕様は可読できる記述で表現されるが、本発明では、これを計算機が処理可能な中間形式に変換して用いる。実際には仕様を取り込むのではなく仕様データを取り込むのが正しいが、誤解の恐れはないため仕様を取り込むは、仕様データを取り込むと同じ意味として使う。

【0028】HW/SW合成部102は、機能・性能仕様101からHWとSWのモデルを合成する手段である。本合成部は、機能解析部102Aと、HW/SW分割・割付部102Bと、最適化部102C、性能解析部102Dと、制約分配（パジェーティング）部102Eと、HWモデル生成部102FとSWモデル生成部102Gと、HW/SW-IPライブラリデータベース・インターフェース部102Hと制御部102Iとから構成される。

【0029】機能解析部102Aは、システムの仕様を解析し、プロセス内の各タスクの設計目標を計算する。入力、仕様のプロセスとその間の起動関係（時間）とプロセスの要求仕様で、出力はプロセスの各種設計目標である。図5は、機能解析部102Aの処理内容を示すフローチャートである。まず、ステップ1で、機能、性能、要求仕様データを取り込む。次に、ステップ2で、各プロセスに与えられた機能・性能要求仕様から、各タスクの目標性能（msecまたはnsec）やプロセスに必要な記憶容量（Bytes）を求める。もし、機能・性能要求仕様が、与えられていない場合は、システムもしくはブロックに与えられている仕様から分配して補間計算し、各タスクの目標性能（msecまたはnsec）やプロセスに必要な記憶容量（Bytes）を求める。更に、ステップ3では、プロセス間の信号ルートやブロック間チャンネルに関するデータフロー量（Bytes/Sec）、起動タイミング（レート：間欠入力の間隔、レイテンシ：開始時刻と終了時刻）を求める。

【0030】HW/SW分割・割付部102Bは、仕様と設計目標から、HWとSWのトレードオフを行って、HW-IPとSW-IPからなるシステム・アーキテクチャを合成する。入力、機能・性能仕様101と機能解析部102Aで求めたタスクの設計目標と要求仕様のアーキテクチャの設計制約〔面積、速度（遅延）、消費電力、SW/HW指定（プロセッサとか通信チャンネル種の指定）、テストコスト〕と、HW/SW-IPライブラリデータベース管理システム103AからHW-IP、SW-IPを入力とする。出力はアーキテクチャ設計データで設計データベース106に格納される。

【0031】割り付けには2つのモードがあり人手介入によるHW/SWのマッピングと自動マッピングである。自動マッピングは第3実施形態で説明する。

【0032】人手マッピングでは、まず、仕様のプロセス内タスクの変換候補 (HW-IP、SW-IP) を IP ライブラリデータベースから取り出した情報から求め、その後アーキテクチャ・テンプレートを使ってどの部品でタスクを実行するかを決め、実際に HDL や C の記述を生成し、性能解析を行い性能が満たされるかどうかをチェックする。

【0033】まず、タスクの HW-IP、SW-IP への変換候補の作成について述べる。仕様変更の可能性の高いタスクに対し設計者が予め SW-IP を選択するこ

- A) プロセッサ+RAM
- B) プロセッサ+RAM+HWブロック
- C) マルチプロセッサ+RAM

次に、ステップ2で、変換候補リストが完成したかどうかを判断する。変換候補リストが完成したら、ステップ3で、すべてのタスクについて変換候補リストが完成したかどうかを判断する。すべてのタスクについて変換候補リストが完成したら、処理が終了する。すべてのタスクについて変換候補リストが完成していない場合には、残りのタスクの1つについて、ステップ1から処理を繰り返す。

【0036】ステップ2で、変換候補リストが完成しない場合には、ステップ4でタスクを細分化し、細分化後のサブタスクに対して設計目標を計算し、夫々のサブタスクを処理を行うべきタスクとして追加し、ステップ1に戻って処理を繰り返す。尚、タスクを実現する HW-IP や SW-IP がいないため、処理の継続が出来ない場合には、適宜、タスクの処理が可能な HW-IP や SW-IP を設計し、HW/SW-IP-DBMS へ登録する必要がある。

【0037】ここで変換に関係の深い、HW/SW-IP ライブラリデータベース管理システム103と、タスク (FNS-IP) から HW-IP、SW-IP への変換方法について詳しく説明する。上で説明した変換ではタスクに対して HW-IP、SW-IP を対応づけているが、上述したようにタスクは定義済みの関数と演算とそれらの集合で表現されており、その関数や演算を以下では FNS-IP と称する。

【0038】図8は、FNS-IP の変換タイプを示す図である。即ち、FNS-IP は以下の4つの変換タイプを持つ。① FNS-IP は、サブ FNS-IP を要素に持つ異なる FNS-IP に置換される。② FNS-IP は、1つまたは複数の HW-IP だけから実現される。③ FNS-IP は、1つまたは複数の SW-IP だけから実現される。④ FNS-IP は、HW-IP と SW-IP の混在として実現される。

【0039】次に、上記4つの対応関係における階層関係を説明する。FNS-IP はソフトウェアのモジュール構成図と同様に複数の IP がそれぞれ複数の階層を持って構成される (ただし、トップの階層がすべての階層

とも可能である。図6は、タスクの HW-IP、SW-IP への変換候補の作成手順を示すフローチャートである。

【0034】最初に、ステップ1でタスクとタスクに与えられた設計目標 (サイズ、遅延、消費電力) を HW/SW-IP ライブラリデータベース管理システムに問い合わせ、変換可能な HW-IP や SW-IP 情報から次のようなタスク1の実現 IP の候補リストを作成する。

【0035】

- {SW-IP(MPU)}
- {SW-IP(MPU)+HW-IP}
- {SW-IP(MPU)+SW-IP(MPU)}

情報をもつのではなく、1つの下の階層だけの情報を持つ)。タスクの実行順序が定義された機能仕様と SW-IP は下位 IP の実行順序と繰り返しの情報も持つ。図9に一例を示す。

【0040】FNS-IP と HW-IP の関係では、複数の HW-IP で実現されるが、下位 IP 間の接続を表現するため、上位 IP (HW-IP0) がある。図10に一例を示す。

【0041】FNS-IP と SW-IP の関係では、モジュールの対応は HW-IP と同じである。したがって対応関係は HW-IP と同じ階層を持つ。違いは実行順番の情報を上位の SW-IP (図11では SW-IP0) が持っている点である。

【0042】FNS-IP と HW-IP と SW-IP の関係は、FNS-IP を通して関係付けられる。図12に一例を示す。

【0043】HW/SW-IP ライブラリデータベース管理システム103の内部では、各 IP はツリー状に関係付けられ、Unix のディレクトリ構造と同様な形式で保存される。図9～図12の階層図から判るように、FNS-IP のディレクトリ内には、FNS-IP0 と HW-IP0 と SW-IP0 と FNS-IP1 のディレクトリがあり、それらはカテゴリ別に、機能 IP、ハードウェア IP、ソフトウェア IP の管理ディレクトリ内の実体、例えば HW-IP0 のディレクトリは HW-IP 管理ディレクトリ内の HW-IP0 ディレクトリが実体でそれとシンボリックリンクされる。

【0044】以上、IP のインスタンスの階層関係を説明したが、次にカテゴリ間の階層関係を説明する。ここでは HW-IP について説明する。上述したように、HW-IP、コミュニケーション IP と記憶 IP の3種があり、それぞれにさらなるクラスが定義されている。処理 IP の下位階層には、CPU、DSP、ASIC、FPGA などのクラスがあり、その CPU の下位階層には標準プロセッサ、構成可変プロセッサなどのタイプがある (図13)。また IP の特性値として遅延、サイズ、消費電力、テストコストの値が最大値、典型値、最小値

に分けられ格納されている。図15に、そのようなHW-IPライブラリデータベースの例を示す。

【0045】変換候補の探索は、HW/SW-IPライブラリデータベース管理システム103Aに対して、FNS-IP名とそのIPの設計目標（サイズ、遅延、消費電力など）をインターフェースして、図9～図13及び図15の構造を持つDB内で条件を満たす候補を探し出す。

【0046】次にHW/SW-IPマッピングの後に行われる、最適化部102Cの最適化機能について説明する。これは初期マッピングされたアーキテクチャのHW-IPとSW-IPを最適化する。入力システム・アーキテクチャ設計データ106とアーキテクチャの設計制約〔面積、速度（遅延）、消費電力、HW/SW指定（プロセッサとか通信チャネル種の指定）、テストコスト〕とHW/SW-IPライブラリデータベース管理システム103AからのHW-IP、SW-IPデータである。出力は、システム・アーキテクチャ設計データ106である。図14は、最適化部102Cの処理内容を示すフローチャートである。まず、ステップ1で、性能解析部102Dを用いて、アーキテクチャの面積、性能、消費電力、テストコストのボトルネック解析を行う。次に、ステップ2で、ネックになっているHW-IP/SW-IPを見つける。次に、ステップ3で、HW-IP/SW-IPの再選択を行う。例えば、CPUの場合はコンフィグレーション・パラメータ（命令セット、ワード数、レジスタ数など）を変更する。また必要に応じて、キャッシュサイズ等を変更する。ステップ4で、再度性能予測を行う。ステップ5で、設計制約を満たしているかどうかを確認する。設計制約を満たしていれば、ステップ6でアーキテクチャ設計データを保存し

て終了する。満足しない場合はステップ2へ戻る。

【0047】次に性能解析部102Dで行われる性能の予測計算を詳しく説明する。これはHW-IPやSW-IPの個々の特性（面積、処理時間、消費電力、テストコスト）を用いて、HW/SW分割後のシステム全体の性能を計算する。入力システム・アーキテクチャ設計データ（HW-IP、SW-IP）と、HW-IP、HW-IPの特性情報（HW/SW-IPライブラリデータベース管理システム103Aから）である。出力は、アーキテクチャ全体の特性予測結果である。処理内容は、アーキテクチャの構成要素であるHW-IPやSW-IPの各種特性を使ってアーキテクチャ全体の特性（面積、性能、消費電力、テストコスト）を静的に計算する。IPの個々の特性は、IP名とそれが持つパラメータをHW/SW-IPライブラリデータベース管理システム103Aに与えて計算して受け取る。性能計算は、アーキテクチャ個々のIPの処理（並列、逐次）を考慮する。たとえば、HW-IP（MPU）での処理時間はSW-IPの実行時間とする。面積の計算では、MPUはHW-IPと同等に扱う。SW-IPのコード量はメモリのサイズの計算に用いる。パラメータが確定していない場合は、上限、下限、典型値を計算する。

【0048】面積は、全HW-IPの面積加算と、HW-IP<sub>k</sub>～HW-IP<sub>1</sub>間のインターフェースHW-IP（バスなどの共有HW: IHW-IP<sub>c</sub>）と、HW-IP相互の個別接続IHW-IP<sub>k-1</sub>（ $k \neq 1$ ）として、HW-IP<sub>k</sub>とHW-IP<sub>1</sub>の接続部分の面積を加算する。またSW-IP格納メモリの面積も付加する。P<sub>1</sub>、P<sub>2</sub>はパラメータを指す。

【0049】

【数1】

$$\sum_{k=1}^n \text{Area}(\text{HW-IP}_k: P_1, P_2, \dots, P_n) + \sum_{c=1}^p \text{Area}(\text{IHW-IP}_c: P_1, P_2, \dots, P_n) +$$

$$\sum_{k=1}^n \sum_{1 \neq k}^n \text{Area}(\text{IHW-IP}_{k-1}: P_1, \dots, P_n) + \sum_{k=1}^m \text{Area}(\text{SW-IP}_k: P_1, P_2, \dots, P_n)$$

性能は、すべてのHW-IP、SW-IPの処理時間の最大値として求める。例えば、ASIC1（HW-IP1）とMPU（HW-IP2）上のソフト（SW-IP1）で実行する場合の処理時間の最大値は、 $\text{Max}(\text{THW}(\text{ASIC1}), \text{TSW}(\text{HW-IP2}, \text{SW-I}$

P<sub>1</sub>)) 消費電力は、HW-IP（MPUやメモリ含む）の消費電力の和として求める。

【0050】

【数2】

$$\sum_{k=1}^n P(\text{HW-IP}_k: P_1, P_2, \dots, P_n) + \sum_{c=1}^p P(\text{IHW-IP}_c: P_1, P_2, \dots, P_n) +$$

$$\sum_{k=1}^n \sum_{1 \neq k}^n P(\text{IHW-IP}_{k-1}: P_1, \dots, P_n)$$

テストコストは、各HW-IPのテスト時間の最大値とテスト時に要する消費電力の積として求める。

【0051】次に、設計制約のバジェット部102Eについて説明する。これは合成されたシステム・ア

ーキテクチャの各HW-IPとSW-IPに対して、設計制約を計算して割り振る。人手による設計制約分配（各種制約の入力コマンドやGUI入力手段）も可能とする。入力は、アーキテクチャ設計データ（HW-IP、SW-IPデータ）とアーキテクチャ全体の特性データ（面積、性能、消費電力、テストコスト）である。出力は各HW-IP、SW-IPまたは、これらを複数含むブロック単位の各種設計制約である。

【0052】まず、本発明の方法の設計後に行われるHW-IPとSW-IP設計処理を述べる。アーキテクチャ確定後、HW-IPは動作合成によってHDLのRTL記述に変換される。SW-IPはコンパイラによってアセンブリ言語、リンクによってサブルーチンの展開が行われる。またアセンブラによって機械語に変換される。従って、HW-IPについては、動作合成やプロセッサ合成の制約として動作周波数、クロック数、面積や消費電力や故障検出率が制約として作成され、SW-IPについてはコード数を制約として作成する。

【0053】性能解析部102Dで得られたアーキテクチャ全体の特性データをもとに、各ブロックの特性（サイズ、遅延、消費電力）の割合を計算し、その割合に基づき、設計制約を各HW/SW-IPに分配する。

【0054】HW/SW-IPライブラリデータベース・インターフェース部102Hは、HW/SW-IPが持つパラメータ（データ幅、レジスタ、メモリサイズなど）を生成する。入力はHW-IPとSW-IPで、出力はHW-IP、SW-IPのパラメータセットである。処理内容はHW-IPやSW-IP（プロセッサ、インターフェースを除く）が持つパラメータ（上述IP内パラメータ）を生成する。また外部環境からもIPの各種パラメータを指定可能である。

【0055】このパラメータと他の処理部分との関係は、性能解析部102Dでアーキテクチャの性能見積もりを行うとき、HW-IPやSW-IPの各種パラメータを固定して各IPの特性をIPライブラリデータベースから取り出す。つまり、生成されたパラメータは特性計算に用いられる。また各種IPのパラメータは確定後、次の設計ステップで利用される。

【0056】HWモデル生成部102Fは、システム・アーキテクチャの設計データベース106から、HWの各ブロックと各ブロックの結線情報を取り出しHWモデル104すなわちHDL記述を出力する。

【0057】SWモデル生成部102Gは、システム・アーキテクチャ設計データベース106からSWの情報を取り出し、さらにSW-IPデータベースからソースデータを取りだしプロセッサで実行されるSWモデル105すなわちプログラムをC言語で出力する。

【0058】制御部102Iは、外部からのコマンド（仕様内タスクのSW割付指示やIPの各種パラメータの指示や分配制約の指定）を取りこんで、その情報を蓄

え必要なサブシステム（処理部）へインターフェースする。また各処理部分が処理に必要な入出力データを用意する。

【0059】以下、全体の処理の典型的な実施例について、処理フローを説明するが、準備としてシステム・アーキテクチャ・テンプレートを説明する。

【0060】システム・アーキテクチャ・テンプレートは、抽象的であつ柔らかなアーキテクチャ・テンプレートで、MPU、DSP、コプロセッサ、グルー論理、バス、制御、メモリなどのクラスやタイプ（上述）からなるHW-IPから構成される。これらは、ユーザがHW/SW-IPライブラリデータベースから各種IPを選択し、また各種インターフェースIPを使って接続して作成される（図16）。

【0061】HW/SWのマッピングは、このテンプレートを使い、上で求めたタスクのHW/SW分配候補リストに基づいて、如何なる種類（後述するクラスやタイプ）のHW-IPやSW-IPで、各プロセスやタスクをいつ実行するか（タイミング）を指定する。ここでのマッピングは、テンプレートにある仮想HW-IPをリストにある具体的なHW-IPに変更する作業でもある。

【0062】図17は、本発明の実施例によるシステム・オン・チップのアーキテクチャ設計支援方法を用いた場合の、処理手順を示すフローチャートである。

【0063】先ず、ステップ1では、システムの仕様を取り込んで解析する。次に、ステップ2では、システムアーキテクチャテンプレートを選択又は作成する。次に、ステップ3では、仕様内の夫々のブロックの実装方法（HWにするかSWにするか）を決める。次に、ステップ4では、ブロック間チャネルを抽象インターフェースIPと対応付ける。テンプレートにチャネル実現のインターフェースIPがない場合は、IPデータベースから取り出し追加する。次に、ステップ5では、仕様内プロセスの実行手順を作成し、各タスクの変換候補リストに基づきHW-IP、SW-IPモジュールを生成する。尚、内部データはあとから生成する。次に、ステップ6で、すべてのタスクに対してIPの生成が終了していなければ、残りのタスクについてHW-IP、SW-IPモジュールを生成する。ステップ6で、すべてのタスクに対してIPの生成が終了していれば、ステップ7で、冗長なIPを削除し、IPのリソースシェアを行う。次に、ステップ8で、プロセスはステートマシンとして、又、ブロックはそれらの記述の呼び出しルーチンとして、HDL記述またはC言語等で各タスクを実装する。次に、ステップ9では、全体の性能解析を行う。次に、ステップ10では、IP割付の最適化を行う。次に、ステップ11では、制約予算の分配を行い、HWとSWの仕様を生成する。次に、ステップ12では、機能検証をシミュレーションで行う。次に、ステップ13

で、シミュレーションの結果に問題があれば、再度処理をやり直す。即ち、チャンネルの問題だけなら、ステップ4に戻る。そうではなくて、一般的な性能の問題ならばステップ3に戻る。ステップ13で、シミュレーションの結果に問題なければ、ステップ14で動作合成、プログラムの最適化を行い処理を終了する。

【0064】HWとSWの分割処理の部分で、図3のSDLのシステムブロックを例に説明する。まず、仕様を解析する。即ち、図7の性能仕様から、タスクの性能目標を計算する。そして、アーキテクチャ・テンプレートを選擇する(図16)。ここで、ブロックB1をSWにブロックB2をHWと仮定する。又、ブロックB1にはプロセスP1とP2が、プロセスP1にはタスクTSK1が、プロセスP2にはタスクTSK2とTSK3が、またブロックB2にはプロセスP3があり、その中にタスクTSK4があるものとする。

【0065】次に、ブロック間チャンネルC1の実現プロトコルとローカルバス(L-Bus)とそのコントローラを対応づける。即ち、テンプレートのインターフェースIPを具象化してその実現IPに置き換える。そして、ブロック内のプロセスの実行順番を性能仕様に基づいて求める。ここでは、P1の後にP2が実行される。

【0066】次に、ブロックB1のプロセスP1データを取り出し、タスク(TSK1, TSK2...)を取り出す。そして、タスクを実現するIP候補(MPU..., MPU+DSP)をHW/SW-IPライブラリデータベース管理システム103から取り出す。以上、すべてのブロックのプロセスのタスクについて同様な処理を行う。その結果、以下のようなIP候補が得られたものとする。

【0067】

TSK1...MPU1, MPU2+COP1

TSK2...MPU2

TSK3...MPU1, SPHW

TSK4...COP1

ここで、COP1はコプロセッサでSPHWは専用HWを示す。このときTSK4がCOP1でしか処理できないことを考えると、以下のようなIPが選擇される。

【0068】

TSK1...MPU2+COP1

TSK2...MPU2

TSK3...SPHW

TSK4...COP1

ここで、TSK3はMPU1で実行可能であるが、性能不足のため専用ハードウェア(SPHW)の実行を選択した。これでタスクのIPへの対応付けが終わったので、ブロックのHDL記述とC言語のプログラムを生成する。図18は、ブロックB1の実行を制御するC言語のプログラムである。なお、ここでは変数、パラメータは省略してある。また図19は、専用HWのVHDL記

述とブロックB2のVHDL記述である。同様に、信号やパラメータは省略している。

【0069】次に、これらのHDL記述とC言語プログラムを入力して性能解析部102Dで性能解析を行う。要求性能と性能解析結果を比較して、要求が満たされない場合はボトルネック解析に基づき割り付けたIPの最適化を行う。プロセッサの可変パラメータを変更して再度性能解析を行い、パラメータの変更を行い、性能をチューニングする。性能が満たされたらHWとSWの制約のパッケージングを行い、HDL記述を生成して終了する。

【0070】(2)第2実施形態

この第2実施形態は、HW-IP、SW-IP人手マッピングサポート機能の特徴とする。以下の説明では、第1実施形態と共通する部分の詳細な説明は繰り返さない。ここでは、システム・アーキテクチャ・テンプレート作成機能と、HW/SW-IP選択機能、SW-IPスケジューリング機能、仕様充足解析機能が提供されている。

【0071】システム・アーキテクチャ・テンプレート作成機能は、設計者がHW/SW-IPライブラリデータベース管理システム103からHW-IPやSW-IPを選び、コンピュータの表示機能のウィンドウ(キャンバス)上にそれらのスキマティックを配置し、そのIP間を接続する機能と、接続途中または接続完了後のアーキテクチャ・テンプレートデータを保存する機能から構成される。

【0072】IPデータベースにはHW-IPとSW-IPが接続されており、図21はその表示例である。第1実施形態でも説明した通り、HW-IP、SW-IPにはクラスがあり、それを頼りに所望のものを索引することができる。図22はクラスのプールにあるIPを表示したもので、このプールから所望のIPを取りだしてキャンバスに配置する。図20は、そこで用いられる設計制約記述を示す図である。

【0073】HW/SW-IP選択機能は、テンプレートにある各SW-IP、HW-IPのさらに具象化されたHW-IP、SW-IPの特性情報の表示と、第1実施形態で作成されたHW/SW選択リストの一つの候補を選んで、テンプレートのIPを具体的なIPに変更する機能を持つ。SW-IP実行スケジューリング機能はSW-IPの実行順番を決める機能である。仕様充足解析機能は、機能仕様の各タスクのすべてがテンプレートの各SW-IPやHW-IPで実現されたかどうか対応関係を洗い出す機能である。

【0074】(3)第3実施形態

この第3実施形態は、HW/SWの自動マッピングを特徴とする。やはり、以下の説明では、第1実施形態と共通する部分の詳細な説明は繰り返さない。

【0075】まず、各種設計制約をパラメータとするコ

コスト関数(式1)を最小化しながら、機能仕様データ  
(タスクとその間のデータ)とタスクの設計目標から機

設計制約のコスト関数:  $F$  (面積、遅延、消費電力、テストコスト) … (式1)

又、その過程で、各タスクを実現する最もよいHW-IPやSW-IPをIP-DBMSから探索して選択し、それらの実行時間に基づいて、リスケジューリングを行い、HW-IPやSW-IPを割り付け、更にSW-IPリソースのシェアを行いシステム・アーキテクチャを生成する。

【0077】これは、第1実施形態のフローを自動化したものに对应し、予めアーキテクチャ・テンプレートを選択しておき、図23のフローチャートに示した処理を自動で行う。

【0078】即ち、ステップ1で、仕様データ(機能、性能、要求:設計制約)と抽象アーキテクチャテンプレートを取り込む。次に、ステップ2で、仕様のブロック間チャネルの仕様に基づいてHW-IP-DBMSからインターフェースIPの情報を取り出しテンプレートの抽象インターフェースIPを具体化的なIPに置き換える。次に、ステップ3で、仕様内プロセスの実行手順を作成(スケジューリング)し、各タスクの変換候補HW-IP、SW-IPモジュールを生成する。やはり、内部データはあとから生成する。次に、ステップ4で、すべてのタスクに対してIPの対応関係の生成が終了していない場合、ステップ3に戻る。終了している場合には、ステップ5で、冗長なIPを削除し、再度スケジューリングしてIPのリソースシェアを行う。次に、ステップ6で、プロセスはステートマシンとして、又、ブロックはそれらの記述の呼び出しルーチンとして、HDL記述またはC言語等で各タスクを実装する。次に、ステップ7で性能解析を行い、ステップ8で制約が満たされていない場合、ステップ9でIP割付の最適化を行い、ステップ6へ戻る。制約が満たされている場合は、ステップ10で、制約予算の分配を行い、HW、SWのモデルを生成する。この後、ステップ11で、動作合成、プログラムの最適化を行い処理を終了する。

【0079】

【発明の効果】(1)ハードウェアもソフトウェアも再利用モジュールを利用するため、システムアーキテクチャ設計の工数が大変小さい。

【0080】(2)サイズ、遅延、消費電力、テストコストを考慮してHW/SWの分割が行えるため、最適なSoCのアーキテクチャが実現できる。

【0081】(3)アーキテクチャ・テンプレートは抽象度が高いため、具象化の選択肢が多く、最適アーキテクチャの設計のため広いスペースで設計できる。

【0082】(4)システムアーキテクチャの設計で使用するテンプレートは再利用モジュールを用いて構成できるため、テンプレート作成の工数が小さい。

能仕様のタスクをスケジューリングする。

【0076】

【0083】(5)システムの仕様は、1つのシステム記述言語を使い状態遷移機械をベースに設計でき、またそれと、ハードウェアとソフトウェアの対応が容易にとれるため、仕様から、システムアーキテクチャの設計のギャップが少なく、また支援ツールの実現も容易である。

【図面の簡単な説明】

【図1】本発明によるシステム・オン・チップのアーキテクチャ設計支援システムの構成図である。

【図2】本発明によるシステム・オン・チップのアーキテクチャ設計支援システムのモジュールの構成を示す図である。

【図3】仕様記述言語による仕様のブロック図である。

【図4】仕様記述言語によるプロセス記述の例を示す図である。

【図5】本発明によるシステム・オン・チップのアーキテクチャ設計支援システムの機能解析部の処理内容を示すフローチャートである。

【図6】本発明によるシステム・オン・チップのアーキテクチャ設計支援方法によるタスクのHW-IP、SW-IPへの変換候補の作成手順を示すフローチャートである。

【図7】本発明によるシステム・オン・チップのアーキテクチャ設計支援方法による性能仕様記述の例を示す図である。

【図8】本発明によるシステム・オン・チップのアーキテクチャ設計支援方法によるFNS-IPの変換タイプの例を示す図である。

【図9】本発明によるシステム・オン・チップのアーキテクチャ設計支援方法によるFNS-IPの階層関係の例を示す図である。

【図10】本発明によるシステム・オン・チップのアーキテクチャ設計支援方法によるFNS-IPとHW-IPの階層関係の例を示す図である。

【図11】本発明によるシステム・オン・チップのアーキテクチャ設計支援方法によるFNS-IPとSW-IPの階層関係の例を示す図である。

【図12】本発明によるシステム・オン・チップのアーキテクチャ設計支援方法によるFNS-IPとSW-IP、HW-IPの関係の例を示す図である。

【図13】本発明によるシステム・オン・チップのアーキテクチャ設計支援方法によるHW-IPのカテゴリの関係の例を示す図である。

【図14】本発明によるシステム・オン・チップのアーキテクチャ設計支援システムの最適化部の処理内容を示すフローチャートである。

【図15】本発明によるシステム・オン・チップのアーキテクチャ設計支援方法によるHW-IPデータベースの例を示す図である。

【図16】本発明によるシステム・オン・チップのアーキテクチャ設計支援方法によるシステム・アーキテクチャテンプレートの例を示す図である。

【図17】本発明の実施例によるシステム・オン・チップのアーキテクチャ設計支援方法を用いた場合の、処理手順を示すフローチャートである。

【図18】本発明によるシステム・オン・チップのアーキテクチャ設計支援方法で生成されたCのプログラムの例を示す図である。

【図19】本発明によるシステム・オン・チップのアーキテクチャ設計支援方法によるHDL記述の例を示す図である。

【図20】本発明によるシステム・オン・チップのアーキテクチャ設計支援方法による設計制約記述の例を示す図である。

【図21】本発明によるシステム・オン・チップのアーキテクチャ設計支援方法によるHW/SW-IPデータベースのIP階層図である。

【図22】本発明によるシステム・オン・チップのアーキテクチャ設計支援方法によるアーキテクチャテンプレートと具象IPの選択の例を示す図である。

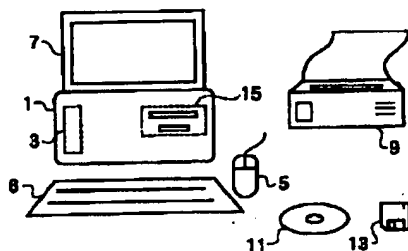
【図23】HW/SWの自動マッピングを行った場合の、本発明の実施例によるシステム・オン・チップのアーキテクチャ設計支援方法による処理手順を示すフロー

チャートである。

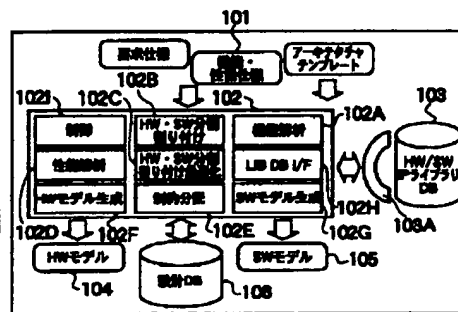
【符号の説明】

- 1 コンピュータ
- 3 内部記憶手段
- 5 マウス
- 6 キーボード
- 7 モニタ
- 9 プリンタ
- 11 CD-ROM
- 13 フレキシブルディスク
- 15 駆動ドライブ
- 101 機能・性能仕様
- 102 HW/SW合成部
- 102A 機能解析部
- 102B 分割割付部
- 102C 最適化部
- 102D 性能解析部
- 102E バジエティング部
- 102F HWモデル生成部
- 102G SWモデル生成部
- 102H インターフェース部
- 102I 制御部
- 103, 103A HW/SW-IPライブラリデータベース管理システム
- 104 HWモデル
- 105 SWモデル
- 106 アーキテクチャ設計データベース

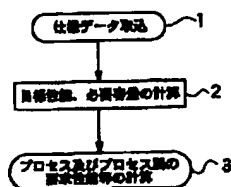
【図1】



【図2】



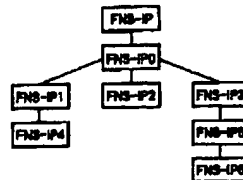
【図5】



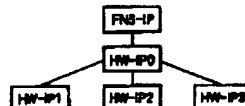
【図8】

FNS-IP = { FNS-IP1, FNS-IP2, ... } — ①  
 = { HW-IP1, HW-IP2, ... } — ②  
 = { SW-IP1, SW-IP2, ... } — ③  
 = { HW-IP1, ... SW-IP1, ... } — ④

【図9】

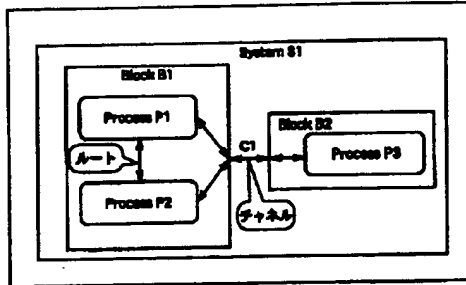


【図10】



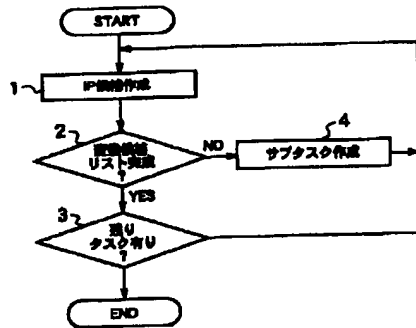
【図3】

【図4】



Process Control:  
 DCL Vin Integer: 整数宣言  
 DCL Vout, Vout\_1 Integer;  
 DCL Const Integer;  
 START;  
 TASK Vout=0, Vout\_1=0, Vin=0;  
 OUTPUT Speed1(0);  
 NEXTSTATE WaitOne;  
 STATE WaitOne;  
 INPUT Control(Const); 入力キューに番号がある場合にトリガ  
 NEXTSTATE WaitSpeed;  
 ENDSTATE;  
 STATE WaitSpeed;  
 INPUT READY1;  
 TASK Vout\_1=Vout;  
 TASK Vout=Vout\_1+Const\*(Vin-Vout\_1);  
 OUTPUT Speed1(Vout);  
 NEXTSTATE WaitSpeed;  
 INPUT SpeedCmd(Vin);  
 NEXTSTATE WaitSpeed;  
 ENDSTATE;  
 ENDPROCESS control;

【図6】

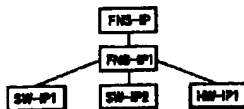


【図7】

【図11】



【図12】



Design Target of S1:  
 Block B1:  
 Process P1:  
 Start Time 100ns;  
 End Time 150ns;  
 Performance from 50MHz to 100MHz;  
 END P1;  
 Process P2:  
 Start Time 50ns after active Vin;  
 End Time 150ns after active Vin;  
 END P2;  
 Memory 1Mbytes;  
 END B1;  
 Block B2:  
 Process P3:  
 Start Time 50ns;  
 End Time 200ns;  
 Performance from 150MHz;  
 END B2;  
 Channel C1:  
 Rate: 100Mbps;  
 End C1;  
 End S1;

【図18】

```

/*ブロックB1の全体制御*/
void main()
{
  P10;
  P20;
}

void P1(...)
{
  task10;
}

void task10
{
  COP10;
}

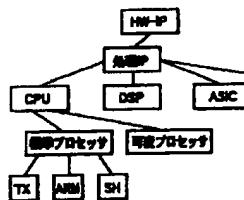
void P20
{
  task20;
  task30;
}

void task20
{
  func0;
}

```

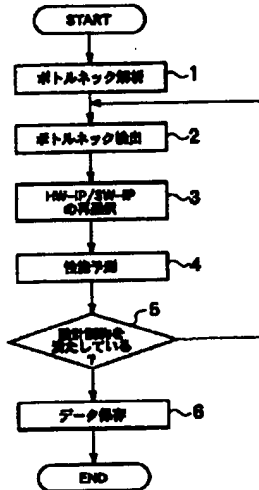
【図13】

【図20】

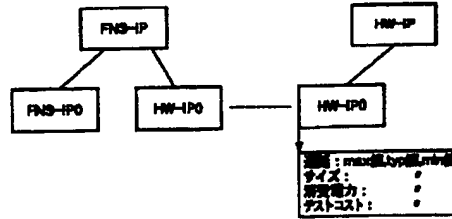


max\_area S1 150000;  
 max\_area SP4W 20000;  
 max\_area B2 30000;  
 max\_delay from A to B 10 ns;  
 max\_delay from C to D 5ns;  
 max\_power S1 2 W;  
 code\_SIZE P1.C 1000;

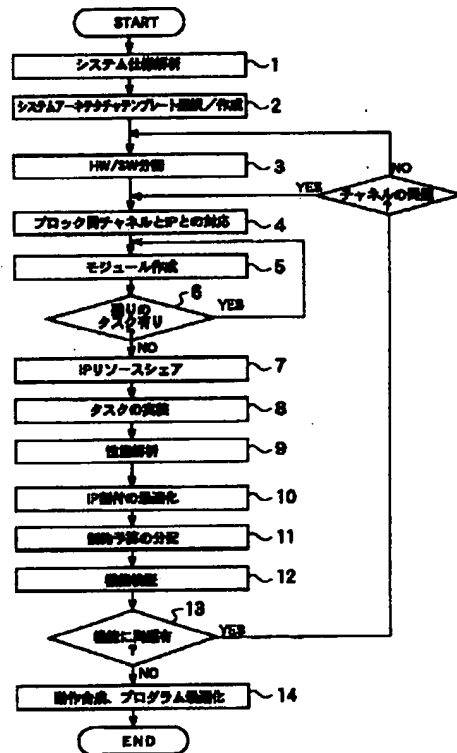
【図14】



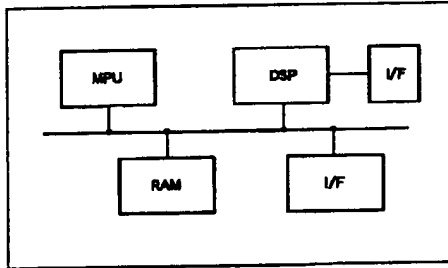
【図15】



【図17】



【図16】

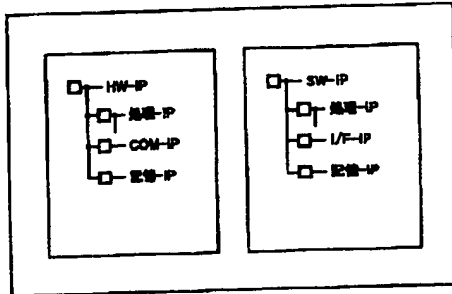


【図19】

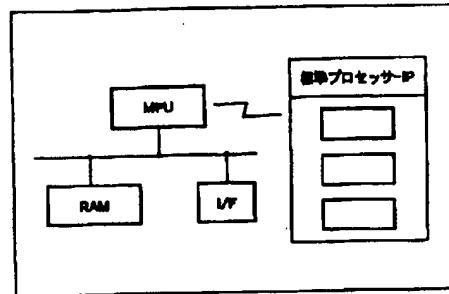
...ブロックB2のVHDL記述  
Entity B2 is  
    ...  
End;  
Architecture BEHAVE of B2 is  
    Begin  
        Process  
        ...  
    End;  
End;

...システム全体のVHDL記述  
Entity S1 is  
    ...  
End;  
Architecture BEHAVE of S1 is  
    Begin  
        ...  
        MPUT1(...);  
        BUS1(...);  
        B2(...);  
    END;

【図21】



【図22】



【図23】

